CMPS 312 Project Phase 1 – UI Design Implementation (20% of the course grade).

The project phase 1 submission is due by 9am Sunday 31th October 2021. Demos will be organized during office hours on the same day.

1. Requirements

You are requested to design and implement **YalaPay** B2B app for managing the payments of invoices. Its purpose is to streamline the financial transactions between a company and its customers by speeding the collection of payments from customers, which has a positive impact on the cash flow of the company and its profitability. YalaPay also manages the cashing cheques and provides valuable financial reports to track pending payments.

Note that storing data in a local database and reading/writing data from remote Cloud Firestore will be done in phase 2 of the project. For phase 1, data should be kept in memory and if needed json files could be used.

The main YalaPay use cases are described Table 1.

Table 1. Use cases description

Use case	Brief description
U1 - Login	Allows the user to login. Login is prerequisite for all the YalaPay use cases shown below.
U2 - Sign Up	Allows the user to create an account to be able to sign in.
U3 - List/Search/Add/Update and Delete a Customer	Provides the ability to List/Search/Add/Update and Delete a Customer. The customer details include: customer Id, company name, address (street, city and country), contract details (first name, last name, mobile and email). When adding a new customer, the customer Id should be auto-assigned.
U4 - List/Search/Add/Update and Delete an Invoice	Provides the ability to List/Search/Add/Update and Delete an invoice. The invoice details include: Invoice number, customer Id, amount, invoice date and due date. When adding a new invoice, the invoice No should be auto-assigned.
U5 - List/Search/Add/Update and Delete payments for an invoice.	Provides the ability to List/Search/Add/Update and Delete payments for an invoice. An invoice can have one or many payments. The payment details include: the payment amount, payment date, mode of payment (cheque, bank transfer or credit card). For cheques more details should be entered: cheque number, due date, cheque image…
U6 - Manage cashing cheques	Manage cashing cheques: - Submit a group of cheques to the bank and set their status to 'Awaiting Caching'. - After few days the user can change verify the cheques status using the company's online banking then update the cheques

	status to either 'Cashed' or 'Returned' and select the Reason for rejection such as 'Insufficient Funds', 'Signature does not match', 'Error in the Cheque'. Note that amount of
U7 –Payments Report	Displays pending or received payments during a period.
U8- Cheques Report	Displays Received/Awaiting Caching/Cached/Returned cheques during a period.

2. Deliverables

Seek further clarification about the requirements/deliverables during the initial progress meeting with the instructor. Note that further important clarifications maybe modified/added to the project requirements.

1) Application design documentation that includes the Repositories Class Diagram.
   During the weekly project meetings with the instructor, you are required to present and discuss your design with the instructor and get feedback. You should only start the implementation after addressing the feedback received about your design.
2) Implement UI for each use case following design best practices. The UI should be fully working using learning package data loaded from a json file. Remember that 'there is elegance in simplicity'!
3) Design and implement the app navigation. It should be fully working, and the user can navigate from one activity to another in intuitive and user-friendly way.
4) Implement the entities and repositories using Kotlin. They should be fully working. Create some test Learning Package data to ease testing. First test them using a main function that displays the results to the console before using them in the UI.
5) Document the testing of UI and repositories using screen shots illustrating the testing results.
6) Every team member should submit a description of their project contribution. Every team member should demo their work and answer questions during the demo.

Push your implementation and documentation to your group GitHub repository as you make progress.

3. Grading rubric

Criteria	%	Functionality*	Quality of the implementation
1) Application Design - Repositories Class Diagram.	5		
2) Design and implement the **UI**	50		
3) Design and implement the **UI Navigation**	10		
4) Implement the entities and repositories using Kotlin	30		
5) Testing documentation using screen shots illustrating the testing of UI and Repositories.	5		
6) **Discussion of the project contribution** of each team member [-10pts if not done]			
Total	100		
Copying and/or plagiarism or not being able to explain or answer questions about the implementation	-100		

* **Possible grading for functionality** - ***Working*** (get 70% of the assigned grade), ***Not working*** (lose 40% of assigned grade and ***Not done*** (get 0). The remaining grade is assigned to the quality of the implementation.

In case your implementation is not working then 40% of the grade will be lost and the remaining 60% will be determined based on of the code quality and how close your solution to the working implementation.

Solution quality also includes meaningful naming of identifiers (according to Android naming conventions), no redundant code, simple and efficient design, clean implementation without unnecessary files/code, use of comments where necessary, proper code formatting and indentation.

Marks will be reduced for code duplication, poor/inefficient coding practices, poor naming of identifiers, unclean/untidy submission, and unnecessary complex/poor user interface design.

Appendix 1 - Example UI design (These screen shots are provided just to clarify the requirements. Surely, they are NOT the recommended design. Be creative and come-up with your own design better than one below).